UML - Unified Modeling Language

Martin Molhanec

Katedra elektrotechnologie, ČVUT - Fakulta elektrotechnická, Technická 2, 166 21 PRAHA 6
e-mail: molhanec@fel.cvut.cz

Abstrakt

UML – Unified Modeling Language je objektově orientovaná visuální metodologie pro modelování rozsáhlých systémů. Jedná se o metodologii vytvořenou předními kapacitami oboru informatiky (Booch, Rumbaugh, Jacobson). Má podporu významných firem z oblasti informatiky (IBM, Oracle, Microsoft, a dalších) a šanci na širokou standardizaci.

Úvod

Přibližně od počátku devadesátých let se datuje rozmach objektově orientovaných metodologií. Nejznámější z nich jsou spojeny se jmény Booch, Rumbaugh, Coad, Jacobson a Yourdon. Nicméně skutečnost, že existuje několik do různé míry rovnocenných metodologií bylo do určité míry současně i brzdou jejich rozšíření. Jako reakce na tento stav byl vznik UML - Unified Modeling Language, jehož stručný popis bude obsahem tohoto příspěvku.

Cíle UML

Motivací vzniku UML bylo vytvoření modelovacího nástroje pro velké průmyslové projekty. Tato nová modelovací technika by měla podpořit vysokou kvalitu projektu a snížit jeho cenu. Měla by být škálovatelná a modulární. V samotném úvodu k UML se jako primární cíle této metodologie uvádějí cíle následující.

1. Poskytnout uživatelovi jednoduchý visuální modelovací nástroj tak, aby uživatel mohl snadno vytvářet a vyměňovat si smysluplné modely.
2. Poskytnout mechanizmus pro rozšiřování a specializaci modelu.
3. Být nezávislí na programovacím jazyku a vývojovém procesu.
4. Poskytnout formalismus pro pochopení modelovací jazyka.
5. Podpořit rostoucí trh s objektově orientovanými technologiemi.
6. Podpora vývojových koncepcí vyšší úrovně, jako jsou například *collaborations, framewors, patterns* a *components.*
7. Integrovat nejlepší dosavadní zkušenosti.

Obsah UML

Celá metodologie je definována několika dobře napsanými dokumenty.

- *UML Summary* - dokument obsahující stručný úvod do cílů a zdrojů metodologie.
- *UML Semantics* - dokument definující sémantiku UML ze tří pohledů
  - *Abstraktní syntaxe*
  - *Dobře navržených pravidel*
  - *Sémantiky*
- *UML Notation Guide* - dokument popisující grafickou notaci UML
- *UML Extensions* - dokumenty popisující extense základního modelu, současné době existují dvě následující
  - *UML Extension for Objectory Process for Software Engineering*
  - *UML Extension for Business Modeling*
- *Object Constraint Language Specifikation* - dokument popisující formální jazyk použitý v UML

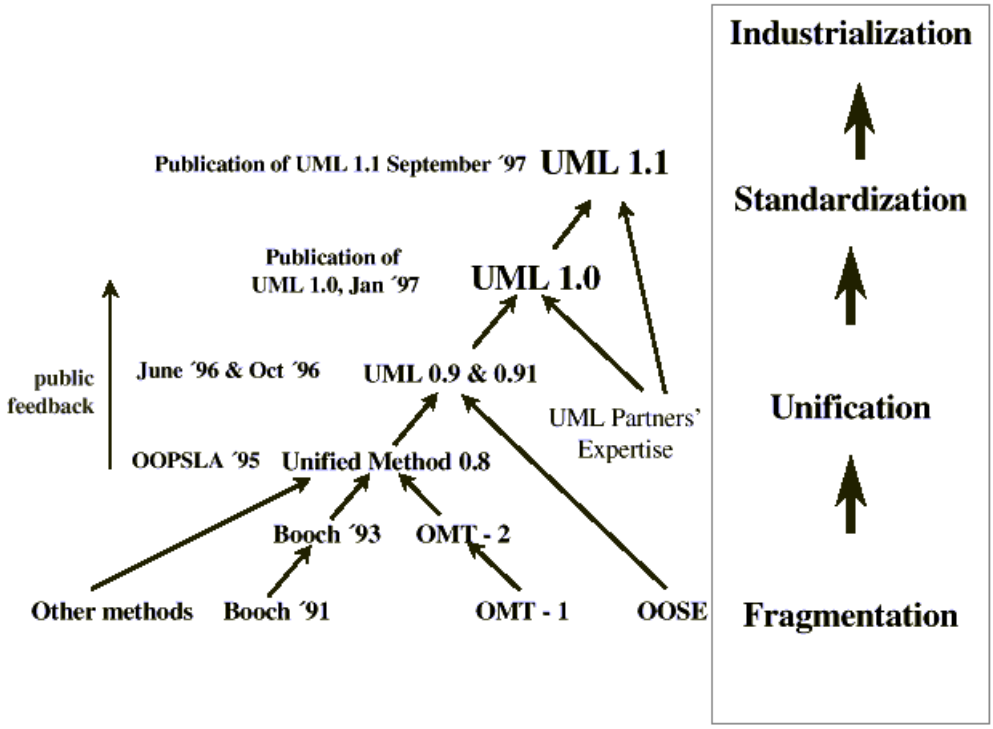
Zdroje UML

Zdrojem UML je celá řada metodologií a metod, uvedeme si jenom některé z nich.

- *Use-case* diagramy podobné diagramům z OOSE
- Diagramy tříd (*Class diagrams*) jsou spojením diagramů z OMT, Booche a dalších.
- Stavové diagramy jsou čerpány od Davida Harela s menšími modifikacemi
- *Activity diagrams* jsou čerpány od fy Oracle a Jima Odella.
- Sekvenční diagramy jsou inspirovány celou řadou před-OO metodologií
- Implementační diagramy jsou od Booche
- OCL (Object Constraint Language) je postaven na Syntropy a Catalysis

Historie UML

Vývoj UML započal v říjnu 1994, kdy Grady Booch a Jim Rumbaugh pracující v Rational Software Corporation započali práci na spojení svých metodologií. Do té doby metodologie Booche a OMT (Object Modeling Technique od Rumbaugh) byly vyvíjeny nezávisle a byly uznávány za vedoucí metodologie v oblasti OOAD (Object Oriented Analyses and Design). První verze UML 0.8 byla vytvořena v říjnu 1995. Koncem roku 1995 se k firmě Rational připojil také Ivar Jacobson a začalo připojování jeho OOSE metodologie. Výsledkem práce těchto velikánů v oblasti OOAD byla verse UML 0.9 koncem roku 1996. Během roku 1996 začalo také strategické spojení, jednak s Object Management Group (OMG), ale také s mnohými komerčními partnery. Výsledkem byla verze UML 1.0 v lednu roku 1997 a koncem roku 1997 verze 1.1, která je v tomto příspěvku popisována. Celkový vývoj UML je naznačen na obr. 1.



Obr. 1 Vývoj UML

Účastníci UML

V současné době se vývoje UML oficiálně účastní mnoho špičkových světových firem. Jedná se o následující firmy Hewlett-Packard, IBM, I-Logix, ICON Computing, IntelliCorp, MCI Systemhouse, Microsoft, ObjectTime, Oracle, Platinum Technology, Ptech, Rational Software, Reich Technologies a Taskon, Softeam, Sterling Software, Unisys.

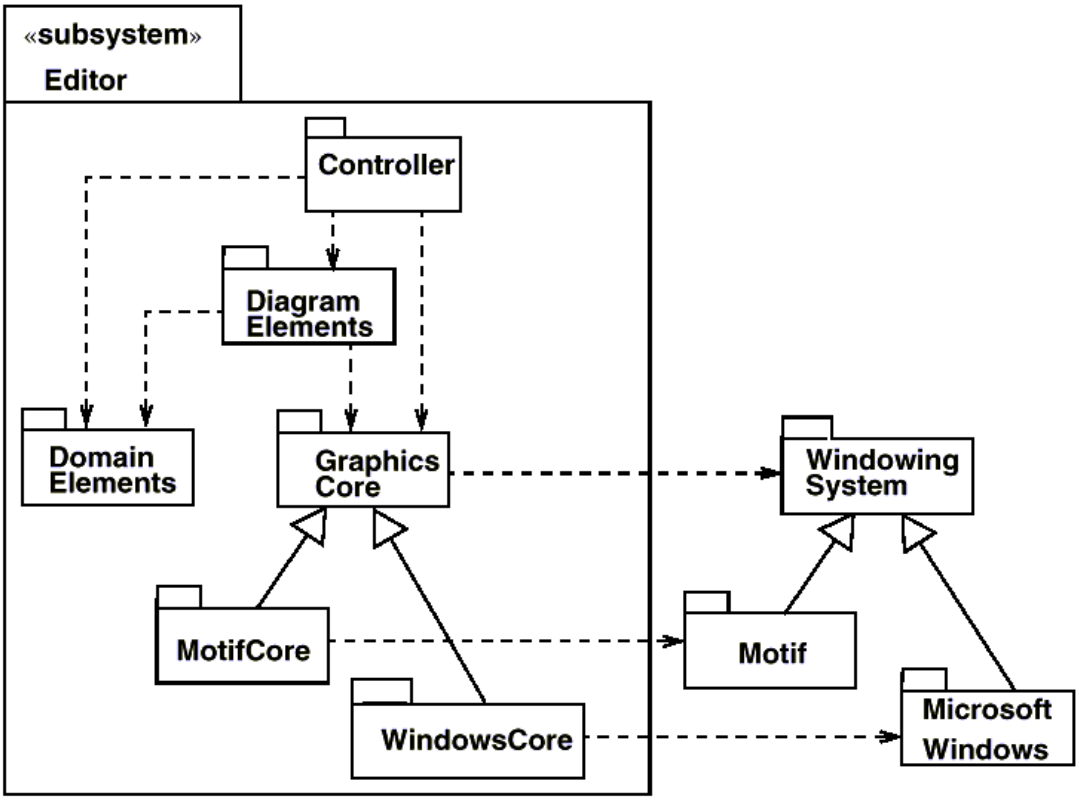
Vývoj UML též podporuje celá řada dalších veličin v oblasti OOAD, například Peter Coad, Mary Loomis, Ed Yourdon a celá řada dalších.

Notace a Sémantika UML

Vzhledem k nesmírné rozsáhlosti UML budu muset v tomto příspěvku seznámit čtenáře jenom s některými základními rysy notace a sémantiky UML. Pokusím se tedy o přehled těch nejdůležitějších konstruktů, které UML poskytuje s tím, že omezím jejich detailní popis. Nejpodrobnější popis se bude týkat diagramu tříd, který je hlavním diagramem většiny OOAD metodologií.

Model Management

Hlavním pojmem této komponenty je *Package*. Package je skupina elementů modelu. Jedná se o nejvyšší úroveň pohledu na celkový model. Ukázka takového diagramu je na obr. 2. Mezi jednotlivými package jsou vyznačeny závislosti. Diagramy této úrovně jsou určeny pro přehledné znázornění vztahů mezi moduly nejvyšší úrovně.

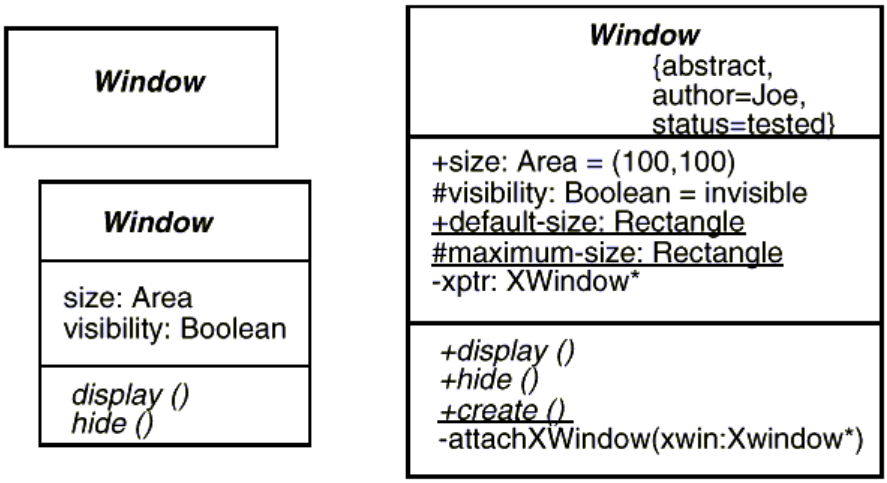


Obr. 2 Package, modul a jejich závislosti

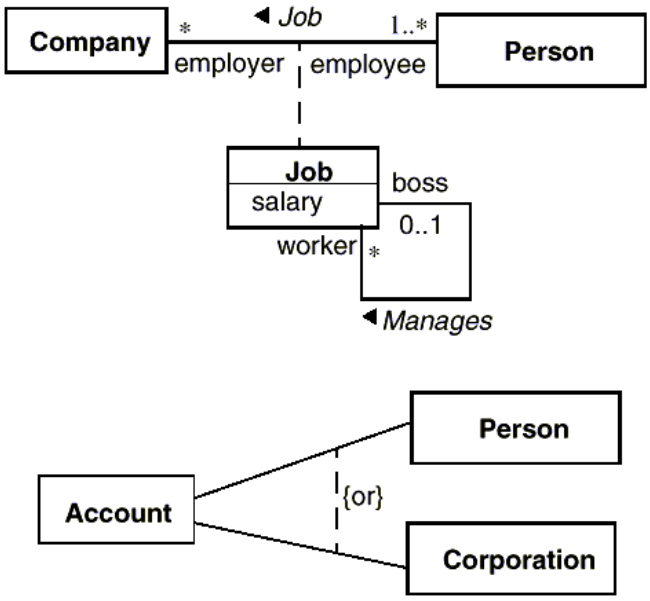
Class Diagram

Jedná se o klasický diagram tříd vycházející z OMT. Je rozpoznávána abstraktní třída a abstraktní metody. Třída obsahuje atributy a metody. Toto dělení je však možné rozšířit na další skupiny. Je také možné označovat viditelnost jednotlivých atributů či metod podobně jako například v jazyce C++. Dále je podporována možnost tříd typu *Interface*, *Template* (parametrizovatelná třída) a *Metatříd*. Ukázka různých zobrazení třídy s atributy je na obr. 3.

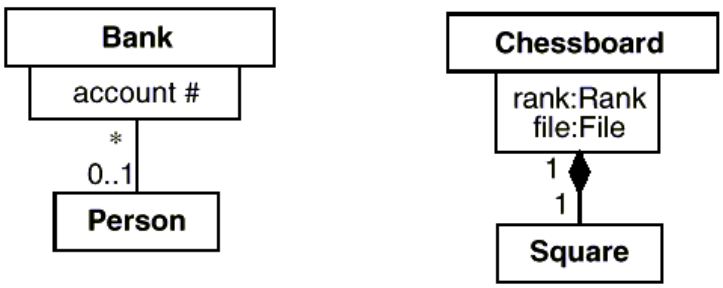
Součástí tohoto typu diagramu jsou též *Associace*, vztahy mezi třídami. Ukázka associace mezi třídami je na obr. 4. UML velice přesně dovoluje stanovovat vlastnosti associace, tj. například jeji *multiplicity* a další vlastnosti. Jsou dovoleny kvalifikované vztahy (obr. 5), vztahy s hodnotami (obr. 4) i N-arní vztahy. Vzhledem k tomu, že UML je objektově orientovaná metodologie, je pochopitelně podporována i dědičnost, a to vztahem *generalization* (obr. 6), u tohoto typu vztahu je možné definovat další kvalifikátory, které lépe specifikují jeho vlastnosti (obr. 7).



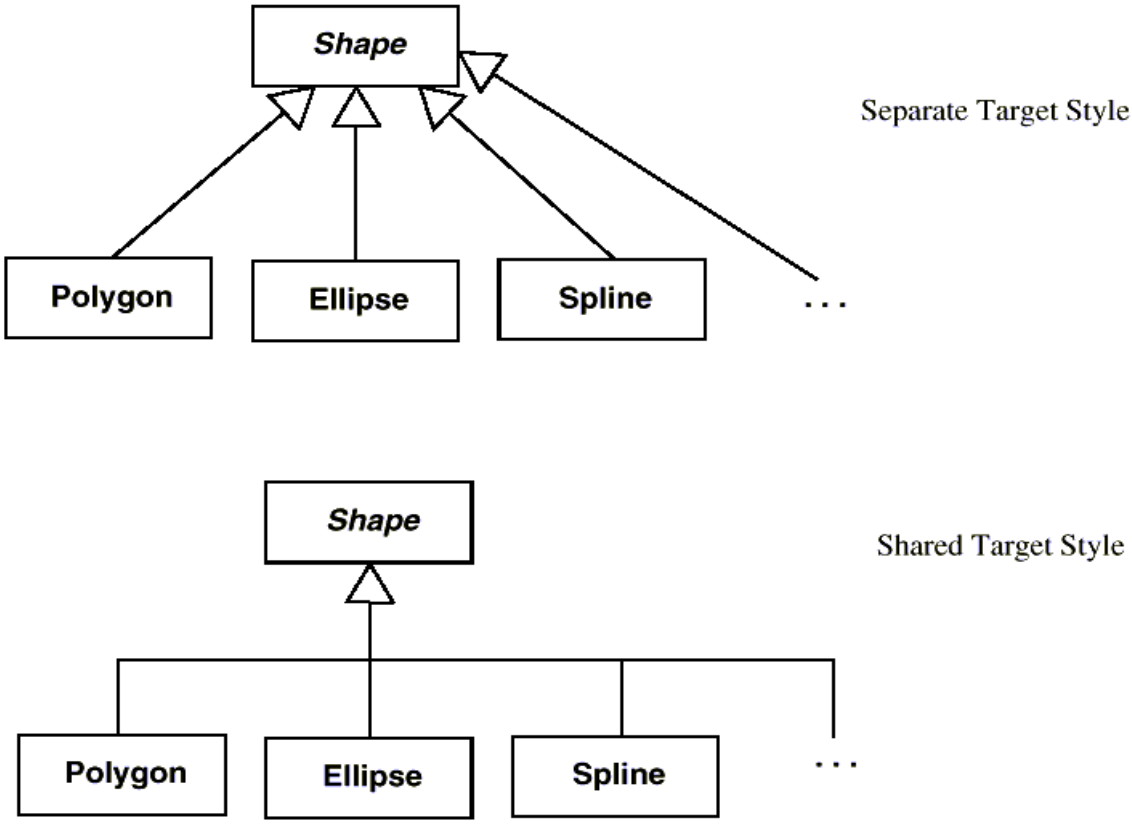
Obr. 3 Varianty znázornění třídy



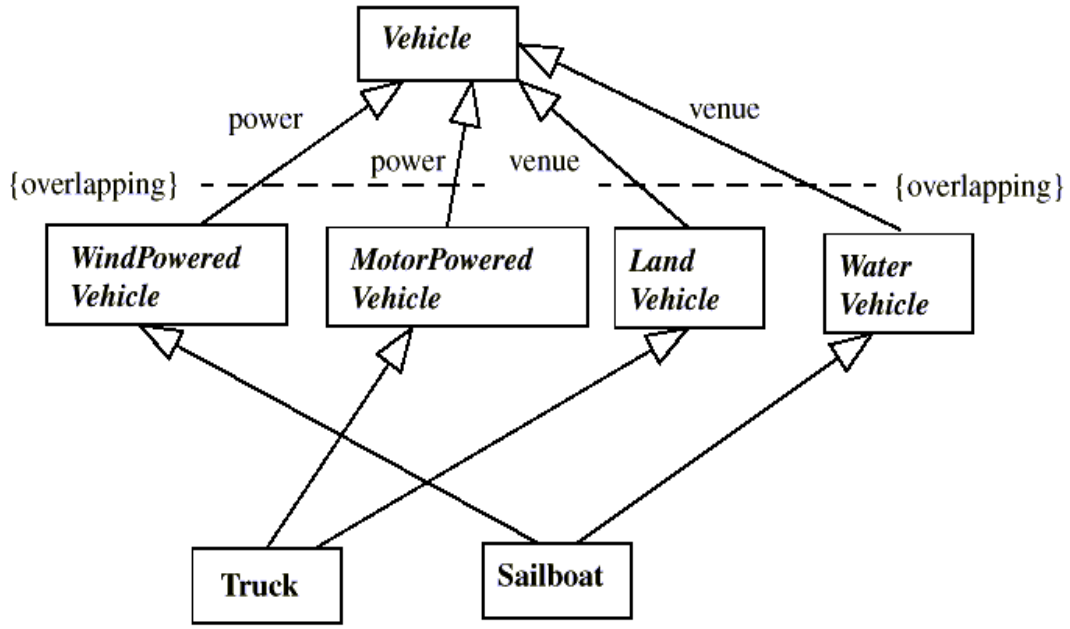
Obr. 4 Různé asociace (vztahy) mezi třídami a vztah s hodnotou



Obr.5 Kvalifikovaný vztah



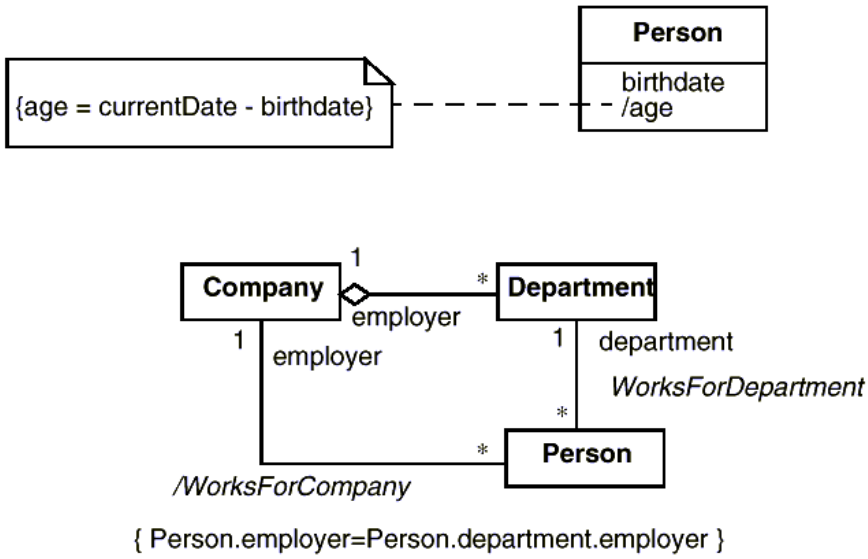
Obr. 6 Vztah generalizace (dědičnosti)



Obr. 7 Další možnosti zápisu vztahu dědičnosti

Mezi rozhodně zajímavé možnosti patří například existence *derived elements*, čili tzv. odvozených (vypočtených) hodnot a172vztahů. Jejich užití je na obrázku (obr. 8). Odvozené elementy jsou uvozeny znakem "\". Diagram tříd je jedním ze

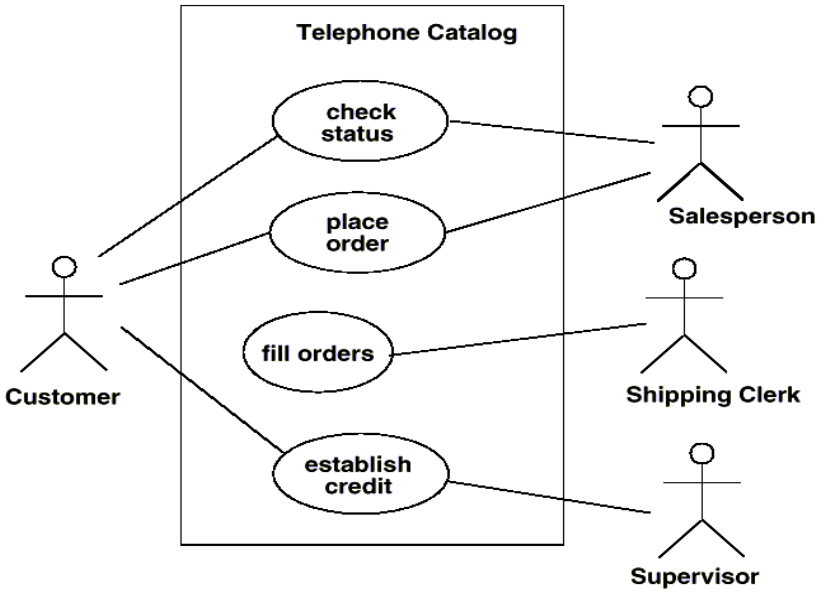
základních diagramů UML a vychází z OMT metodologie.



Obr. 8 Odvozené atributy a vztahy

Use Case

Use Case diagram vyjadřuje vztahy mezi *Actors* vně systému a *Use Case* uvnitř systému. Příklad jednoduchého Use Case diagramu je na obr. 9. Podle mého názoru je tato část UML nejméně rozpracovaná část celé metodologie, lze v ní snad nalézt určitou analogii s DFD nejvyšší úrovně.

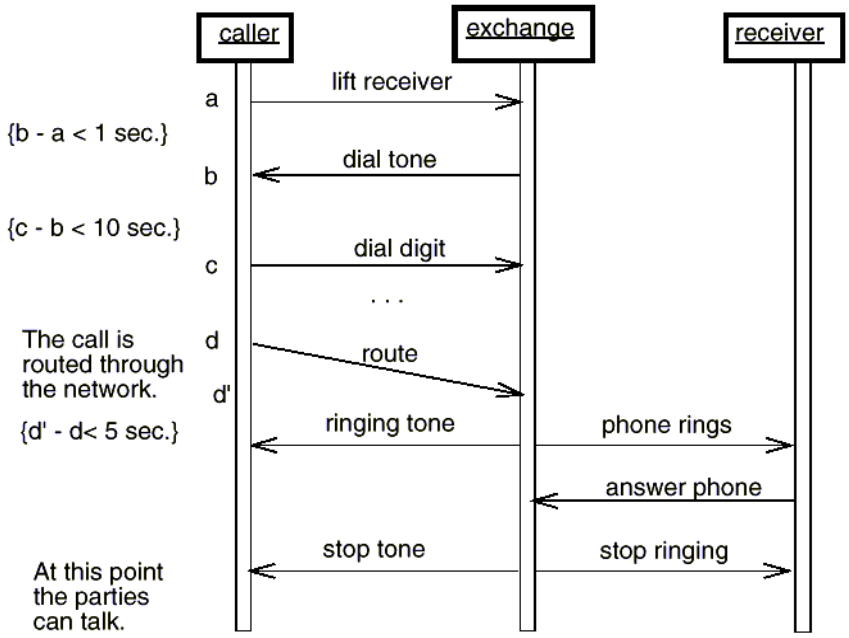


Obr. 9 Use Case diagram

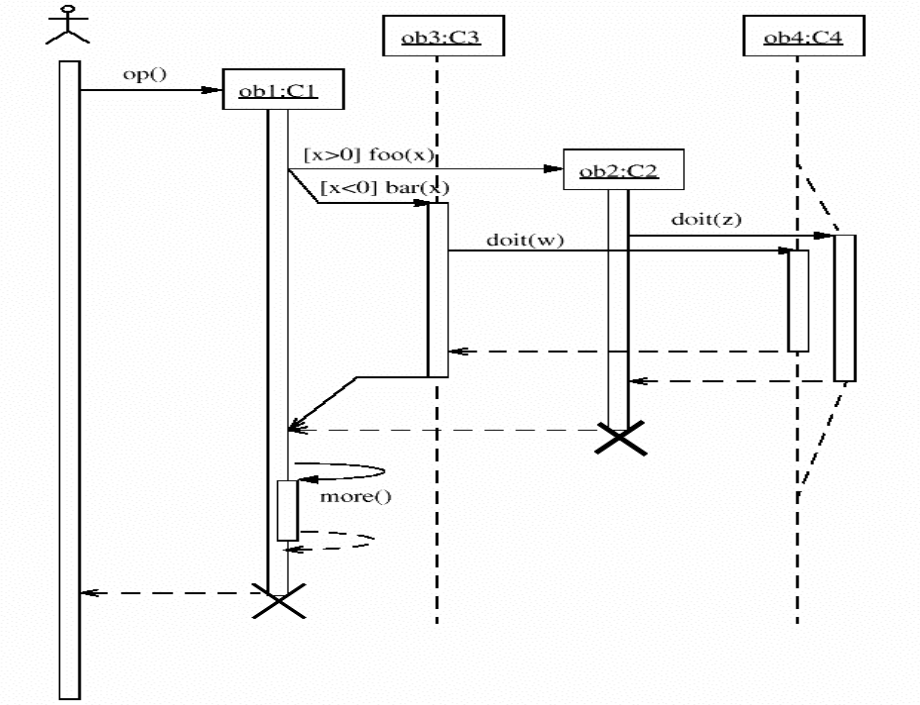
Sequence

Sekvenční diagram popisuje interakci mezi jednotlivými objekty systému v závislosti na čase. Sekvenční diagram má dvě dimenze. Vertikální osa představuje čas a na horizontální ose jsou zobrazeny různé objekty. Čas plyne ze shora dolu. Měřítko na časové ose není normálně zajímavé vyjma real-time systémů. Pořadí objektů

na vodorovné ose není významné. Ukázky dvou sekvenčních diagramů jsou na obr. 10 a 11.



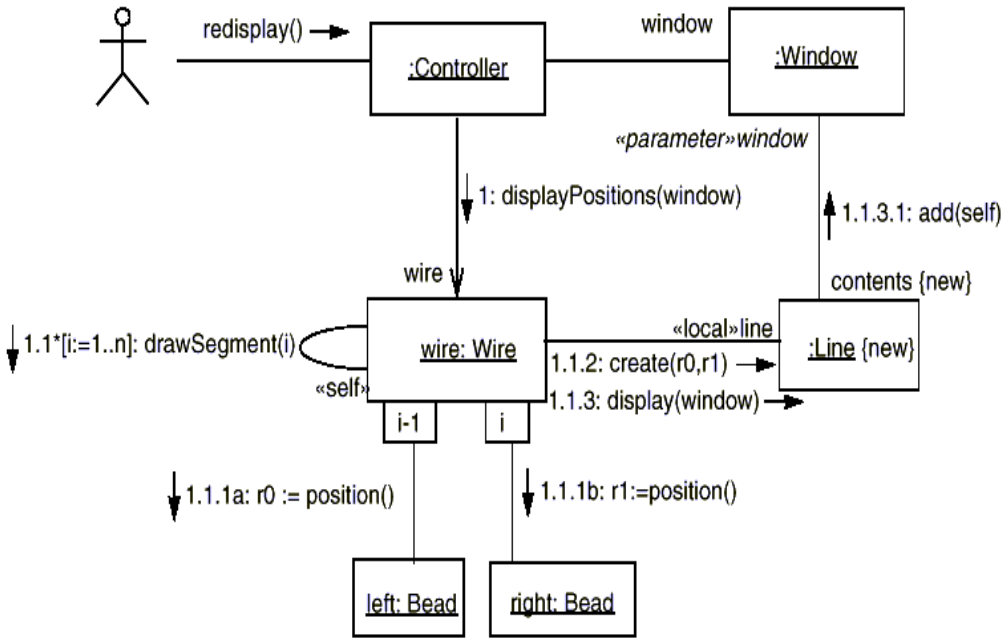
Obr. 10 Sekvenční diagram, první varianta



Obr. 11 Sekvenční diagram, druhá varianta

Collaboration

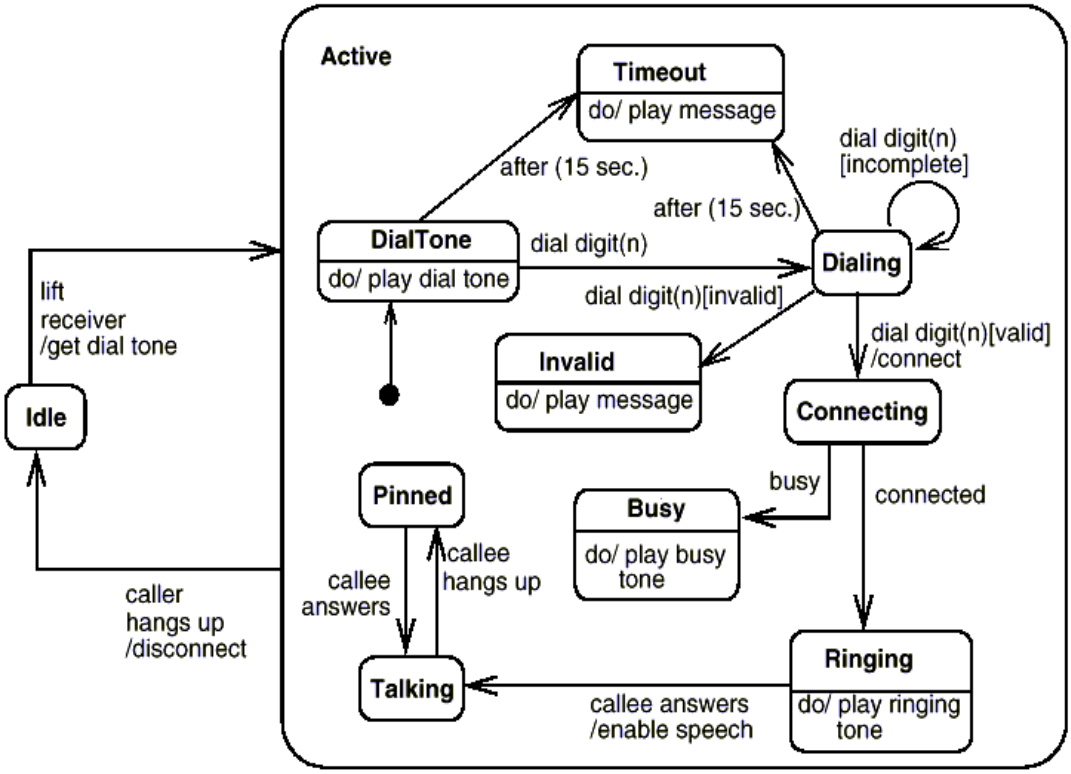
U tohoto diagramu jsem na rozpacích, jak správně přeložit jeho název. Budu používat název diagram spolupráce. Diagram spolupráce zobrazuje interakci mezi objekty a jejich vzájemné vztahy. Ukázka diagramu spolupráce je na obr. 12. Důležitou roli na diagramech spolupráce hraje označování vztahů (zpráv) mezi objekty a jejich přesná interpretace.



Obr. 12 Diagram spolupráce

Statechart

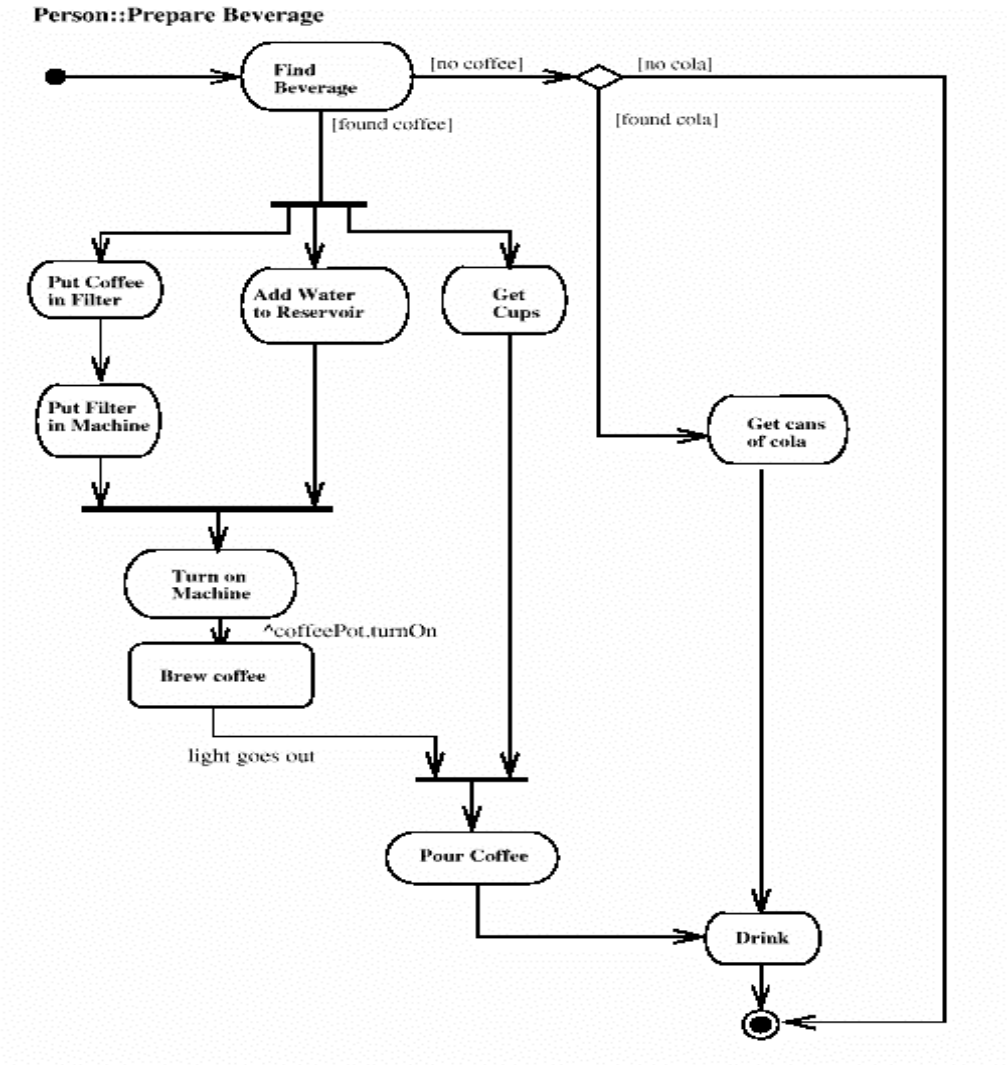
Jedná se stavové diagramy s velice přesně vyjádřenou sémantikou, jak je patrné z obr. 13. Dle UML je *stav* část života objektu během které splňuje některé podmínky, činí některé akce nebo čeká na nějakou událost. Objekt může zůstat v určitém stavu po určitou dobu. Existuje jeden počáteční a libovolný počet koncových stavů. UML metodologie podporuje vnořovaní stavů, čili určitý stav je možné rozkreslit opět jako nový stavový diagram. Přechod ze stavu do stavu je atomický, není ho možné dále dělit.



Obr. 13 Stavový diagram

Activity

Diagram *aktivit* je podobný diagramu stavovému. Současně se podobá například vývojovému diagramu (obsahuje rozhodovací bloky) nebo diagramu Petri-Nets (obsahuje synchronizační body). Nejlepší představu si čtenář udělá z jeho ukázky (obr. 14).



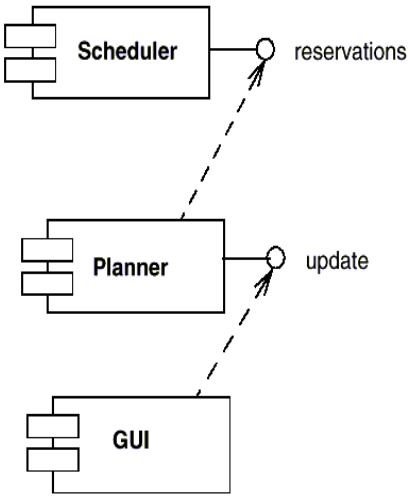
Obr. 14 Diagram aktivit

Component

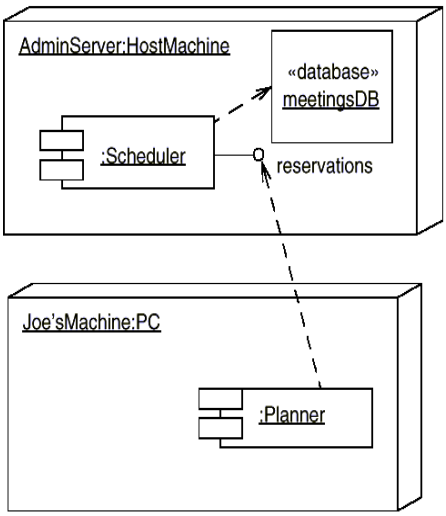
Diagram komponent zobrazuje vztahy mezi softwarovými komponentami, včetně zdrojových kódů, binárních knihoven i spustitelných programů. Jednoduchá ukázka je na obr. 15.

Deployment

Jedná se o diagram konfigurace softwarových komponent během chodu systému. Podobně jako předchozí diagram je vkladem Booche do UML. Jednoduchá ukázka je na obr. 16.



Obr. 15 Diagram komponent



Obr. 16 Deployment diagram

PCL - Object Constraint Language Specification

Většina modelovacích technik nedokáže všechny závislosti mezi jednotlivými konstrukty vyjádřit pouhým grafickým znázorněním a musí si pomáhat popisem. Slovní popis však není vždy jednoznačný, proto byl pro použití v UML vyvinut speciální formální jazyk *Object Constraint Language (OCL)*, jednoduchý pro zápis i čtení. Zdrojem toho jazyka byl obchodní modelovací jazyk interně používaný ve firmě IBM.

OCL není programovací jazyk, je to jazyk určený pro modelovací techniky. Jedná se však o typový jazyk. K jakému účelu je ho možné v UML použít? Například pro specifikování podmínek vykonání operací či metod, pro specifikování invariantů tříd, jako navigační jazyk, pro definování constraint.

OCL umí pracovat s množinami objektů, například s *collection, set, bag, sequence*. Jako ukázku vyjadřovacích schopností jazyka OCL si uvedeme výraz, který má vyjádřit podmínku, že všechny instance osoby mají rozdílne jméno.

Person.allInstances→forAll(p1, p2 | p1 <> p2 implies p1.name <> p2.name)

Person.allInstances je *set* (množina) všech osob a je typu Set(Person). Nad touto množinou je aplikována operace forAll, která vykoná definovanou činnost pro všechny prvky množiny. Parametrem činnosti jsou dva prvky množiny, pro které platí, že pokud jsou rozdílné, mají rozdílná jména. Je vidět, že definování podmínky je deklarativní a nikoliv algoritmické.

Dalším příkladem je podmínka, která vybere všechny zaměstnance jejichž věk, je větší než v 50 let.

self.emloyee.select(p : Person | p.age > 50)

Tento výraz čteme následovně. Na množinu zaměstnanců aktivní třídy je uplatněna operace výběru (select), která vybíra prvky splňující podmínku, že věk zaměstnance je větší než hodnota 50.

Dalším příkladem je výraz jehož výsledkem je množina všech rozdílných dat narození všech osob.

self.emloyee→collect(birthDate)→asSet

Na množinu zaměstnanců je uplatněna operace *collect*, která vytvoří množinu všech dat narození, nad touto množinou je dále uplatněna operace *asSet*, která z této množiny vytvoří množinu všech rozdílných dat narození všech zaměstanců.
Doufám, že výše uvedené ukázky naznačí možnosti jazyka OCL, více než pokus o neucelený popis jeho syntaxe a sémantiky.

Závěr

Vytvoření UML je vítaným přínosem do standardizace analytických metod. Jeho masivní podpora může mít pozitivní vliv na rozšíření nástrojů pro podporu této metodologie. Na straně druhé se zdá, že otevření UML směrem k jeho rozšiřování může být určitou zárukou pro to, aby se nejednalo uzavřený systém, který by nebyl schopen akceptovat nové poznatky a koncepce v oblasti analytických a návrhových metod.

Nicméně se mi zdá, že některé části UML jsou velice propracované, například diagram tříd, stavový diagram a některé další, se zdají být spíše jejím určitým přívažkem, například *Use Case* diagram nebo diagram komponent. Lze očekávat, že vývoj UML bude nadále probíhat a že jeho současný stav není zřejmě ještě definitivní. Zajímavá je také skutečnost, že v metodologii UML není zahrnut známý Data Flow Diagram (DFD). Podle autorů UML není potřeba, protože je nahrazen jinými diagramy obsaženými v UML, například diagramem stavovým, spolupráce, aktivit atp. Nicméně já osobně o této skutečnosti nejsem zcela přesvědčen.

Přední společností, která vyvíjí nástroje CASE s UML metodologií je pochopitelně společnost *Rational Software*, pro kterou pracují samotní autoři této metodologie. Specifikace UML metodologie je dostupná ve formě .PDF souborů včetně evaluační verze CASE nástroje této firmy. Produkty firmy Rational Software v naší republice distribuuje firma Unicorn Distribution se sídlem v Praze. Školení na CASE produkty firmy Rational poskytuje firma OK System se sídlem v Praze, od které lze (aspoň mne se to povedlo ☺) získat CD disk s výše uvedeným obsahem (popis UML a evaluační verse CASE od firmy Rational). Významným hnacím motorem pro UML může být také skutečnost, že některé části CASE nástrojů firmy Rational Software jsou obsaženy ve Visual Studiu firmy Microsoft.

Doufám, že příští rok bude s metodologií UML více zkušeností zejména praktických a budeme si o ní moci sdělit více dalších zajímavých informací.

Odkazy

Rational Software .....http://www.rational.com
UML .....http://www.rational.com/uml
OCL .....http://www.software.ibm.com/ad/ocl
Unicorn Distribution .....http://www.unicorn.cz/distribution
OK System ..... http://oksystem.cz
FAQ about UML.....http://microgold.com/Stage/UML_FAQ.html
UML Resource Center ..... http://www.krumbach.de/home/jeckle/unified.htm